1050-Floppy mit Happy-Enhancement

Wie die Sache arbeitet und wie man die neuen Möglichkeiten nutzt, lesen Sie im ersten Teil unseres Kurses

Sicher weiß jeder, daß eine 1050-Diskettenstation, die mit einem Happy-Enhancement ausgerüstet ist, eine Vielzahl neuer Möglichkeiten bietet. Wie man diese nutzen kann, soll hier gezeigt werden.

Diese Station ist im Grunde ein kleiner Computer, der nur

8 Bit

dazu dient, die Daten sicher auf Diskette zu schreiben und zu lesen. Ihr Prozessor ist ein 6502, der mit 8-KByte-ROM und 8-KByte-RAM den Datenaustausch zwischen Computer und Floppy-Disk-Controler (FDC) regelt. Ferner ist noch ein RIOT 6532 enthalten, dessen Aufgaben später besprochen werden.

Aus mir unbekannten Gründen besteht das Floppy-ROM nicht aus einem 8-KByte-Block, sondern es ist in zwei 4-KByte-Blöcke unterteilt, die den Adreßbereich $fff0 bis $ffff belegen. Das Umschalten zwischen beiden 4-KByte-Blöcken geschieht durch Load-Befehle:

LDA $fff8	Blendet den ersten 4-KByte-Block bei Adresse $f000 bis $ffff ein.
LDA $fff9	Blendet den zweiten 4-KByte-Block bei Adresse $f000 bis $ffff ein.

Für ein problemloses Umschalten zwischen beiden Blökken werden gleich beim Einschalten der Floppy zwei Routinen ins RAM kopiert:

JSR $9600 .WORD adr	Springt in den anderen Block nach adr.
JSR $960A .WORD adr	Ruft ein Unterprogramm im anderen Block bei adr auf.

Wenn man vom ersten Block in den zweiten springen will, so muß die Zieladresse im zweiten um $8000 vermindert werden. Dies sagt den Routinen im RAM, aus welchem Block man kommt und in welchen man will. Das Unterprogramm zur Berechnung von Spur- und Sektornummer befindet sich z.B. in Block zwei bei Adresse $f7db. Will man es aus Block eins aufrufen, so schreibt man:

```
JSR $960A
.WORD $77db
```

Der Adreßbereich des RAM liegt bei $8000 bis $9fff, ferner auch von $0000 bis $01ff. Auf den Bereich zwischen $9800 und $9fff kann nicht ohne weiteres zugegriffen werden. Den RAM-Bereich zwischen $8000 und $97ff spricht man an, indem ein Read- oder Write-Kommando mit der RAM-Adresse als Sektornummer ausgeführt wird.

Der Floppy-Controler ist von Western Digital und hat die Bezeichnung WD 2793-PL 02. Er kann über folgende Register angesprochen werden:

$400 Kommando-Register (STA $400)
$400 Status-Register (LDA $400)
$401 Spur-Register
$402 Sektor-Register
$403 Daten-Register

Die einzelnen Kommandos des FDC:

$88 Lese Sektor
$a8 Schreibe Sektor
$c0 Lese Adresse
$e0 Lese Spur
$f0 Formatiere Spur
$d0 Erzwinge Interrupt

Der RIOT 6532 enthält einen kompletten PIA 6520, wie er auch im Atari enthalten ist (Joystickports). Dazu kommt noch ein Timer.

Die Adressen des RIOT:

$280 PORTA
$281 PACTL
$282 PORTB
$283 PBCTL

Die Funktion der einzelnen Bits:

PORTA:

7 1 FDC fordert Daten an.
  0 FDC fordert keine Daten an
5 1 FDC im Single-Density-Modus
  0 FDC im Double-Density-Modus
4 1 Schreib-Vorkompensation ein
  0 Schreib-Vorkompensation aus
3 1 Motor aus
  0 Motor ein

Bit 0 und 1 werden zur Festlegung der Laufwerknummer benützt.

PORTB:

7 1 Computer sendet gerade Kommando

Meister der Disketten

Benötigen Sie einen professionellen Programmierschutz für Ihre Software? Kein Problem mit dem neuen Programm "Diskmaster", das wir Ihnen hier vorstellen.

Ohne Zweifel übt das Thema Kopierschutz auf viele Leute eine enorme Faszination aus. Mit dem Programm "Diskmaster" ist es für jedermann möglich, einen Kopierschutz zu erzeugen, der selbst professionellen Ansprüchen genügt. Voraussetzung ist ein 1050-Laufwerk mit einer Happy-Erweiterung bzw. ein dazu kompatibles Produkt.

Erstaunlicherweise kann "Diskmaster" mit diesem Hardware-Zusatz sogar Kopierschutzformate erstellen, die sich nicht einmal mit der Happy-Erweiterung kopieren lassen, geschweige denn mit einem normalen Sektorkopierer.

8 Bit

Der zentrale Teil des Programms ist der sogenannte Format-Editor, mit dem ein Kopierschutzformat in einer Art eigener Programmiersprache entworfen werden kann. Dabei läßt sich jedes einzelne Byte des Formats bestimmen; mehr oder weniger kann der FDC (der Floppy-Controller der 1050-Station) direkt angesprochen werden. Wer sich schon einmal mit der Floppy-Anschaltung beim Atari beschäftigt hat, weiß, daß dies keine einfache Sache ist. Der FDC gehorcht nämlich nicht den Befehlen des Mikroprozessors im Computer, sondern verfügt über eine eigene CPU, die lediglich über einen seriellen Bus mit der zentralen CPU in Verbindung steht. Zudem ist das normale Betriebsprogramm der Floppy-CPU penibel dazu ausgelegt, solche nicht vorgesehenen Aktionen strengstens zu verhindern.

Durch den Format-Editor lassen sich Kopierschutzformate mit doppelten Sektoren oder absichtlichen Lesefehlern erzeugen oder einfach mehr Sektoren als gewöhnlich auf einer Spur unterbringen. Nach Wunsch kann dabei der Single- oder Double-Density-Modus verwendet werden. Dennoch ist es nicht ganz leicht, einen Kopierschutz zu erstellen, da man schließlich jedes Byte auf der Diskette berücksichtigen muß. Glücklicherweise enthält die Rückseite der Diskette noch eine Reihe fertiger Formate als Files, die man sofort ausprobieren und nachvollziehen kann. Außerdem sind auch einige Basic- und Assembler-Programme zu finden, welche die Abfrage der verschiedenen Kopierschutzformate als Beispiel zeigen.

Neben dem Editor besitzt "Diskmaster" noch einige weitere Funktionen, die zum Speichern und Laden von Formaten und sonstigen Daten dienen. Natürlich darf auch eine Option zum Aufbringen des Formats

Happy-kompatible Erweiterung erforderlich

nicht fehlen. Erwähnenswert ist noch die Funktion SPUR EINLESEN; sie überträgt eine ganze Spur mit allen darauf befindlichen Daten, also auch Sektor-Header und Prüfsummen-Bytes, in den Speicher des Computers.

Die sehr ausführliche Dokumentation umfaßt 15 DIN-A4-Seiten. Sie ist – was ich für besonders wichtig halte – nicht nur auf die Bedienung des Programms zugeschnitten, sondern erklärt auch einige wichtige Grundlagen der internen Funktion der Diskettenstation. Das ist gerade bei der Erzeugung eines Kopierschutzes von großer Bedeutung, da zu diesem Zweck alle Eigenheiten der Floppy genutzt werden. Allerdings ist es natürlich unmöglich, eine so komplexe Thematik dem Laien auf wenigen Seiten erschöpfend zu erklären; einige Vorkenntnisse sind hier schon erforderlich. Wer mit Begriffen wie Gapbyte und CRC-Generator nichts anzufangen weiß, wird bei der Lektüre schnell überfordert sein.

Trotzdem ist "Diskmaster" mit seinem günstigen Preis von 24.90 DM allen zu empfehlen, die mehr über ihr Diskettenlaufwerk und seine Funktionsweise wissen möchten. Beim Kauf ist unbedingt zu beachten, daß man zur Anwendung des Programms zumindest eine Happy-kompatible Erweiterung braucht. Funktionsähnliche Produkte wie Speedy 1050 oder Turbo 1050 sind hier nicht verwendbar. Im Zweifel fragen Sie am besten Ihren Händler.

Zu haben ist das Programm beim Software-Versand des **ATARI***magazins* (Bestellschein S. 29).

Peter Finzel

0 Computer sendet kein Kommando

6 Dateneingang vom seriellen Bus

Bit 2 bis 5 werden für den Stepmotor benützt.

0 Datenausgang auf seriellen Bus

Da der FDC für bestimmte Kommandos (Formatiere Spur, Lese Spur und Lese Adresse) einen Index-Impuls vom Floppy-Disk-Interface benötigt und dieser Impuls nicht hardwaremäßig geliefert wird, verwendet man hierfür den Timer. Dieser gibt an den FDC einen Index-Impuls ab, sobald er auf Null abgelaufen ist. Die Register des Timers:

$294 LADTIM Liefert Momentanwert des Timers

$296 FASTIM Ändert die Abzählgeschwindigkeit des Timers

$29f STATIM Setzt den Anfangswert des Timers

Der Index-Impuls wird vom Timer geliefert

Unser erstes Ziel soll es sein, ein Listing des Floppy-ROM anzufertigen. Hierzu müssen wir der Floppy einen neuen Befehl beibringen. In ihrem RAM befindet sich bei Adresse $97a0 eine Tabelle, die alle Befehlssymbole, die der Floppy bekannt sind, enthält. Die Startadressen der zugehörigen ROM-Routinen sind in zwei weiteren Tabellen bei Adresse $97c0 (niederwertiges Byte der Startadresse) und $97e0 (höherwertiges Byte der Startadresse) enthalten. Bei einer Happy mit installiertem U.S.-Emulator sieht das folgendermaßen aus:

$97a0 .BYTE "PWpwRrS!"?$NOHQ"
Dann folgen 17 Nullen.

$97c0 .BYTE $17, $17, $12, $12, $76, $71, $93 bis ..., $82, $7a

$97E0 .BYTE $78, $78, $78, $78, $72, $72, $f7 bis ..., $76, $76

Man kann daraus ablesen, daß z.B. die Status-Routine in ROM-Block eins bei Adresse $f793 und die Q-Routine in ROM-Block zwei bei Adresse $f67a beginnt.

Wenn man der Floppy nun einen eigenen Befehl beibringen will, so muß man sein Symbol in die Tabelle ab Adresse $97a0 und seine Adresse in die Tabellen ab $97c0 und $97e0 schreiben. Platz für Ihre eigenen Befehle finden Sie im RAM von Adresse $8300 bis $95ff. Nun muß der Befehl ins RAM der Floppy übertragen werden, das noch vor Zugriff durch das Betriebssystem geschützt werden muß.

Eine Spur wird bei normaler Betriebsart auf einen Sitz in das RAM eingelesen, und jeder Sektor, der danach auf dieser Spur gelesen werden soll, wird direkt aus dem RAM zum Computer übertragen. Dies beschleunigt das Laden von Programmen um ca. 20%.

Ist die Floppy jedoch programmiert worden, so würde dies den Speicherinhalt wieder verändern und den neuen Befehl zerstören. Deshalb läßt sich das RAM durch den Befehl H (Status 0, DAUX $6060) vor einem Zugriff des Betriebssystems schützen.

Zu beachten ist außerdem, daß ein Befehl als Unterprogramm aufgerufen wird und somit über eine RTS-Instruktion verlassen werden muß.

Ein Programm, das alle diese Aufgaben übernimmt, finden Sie in Listing 1. Es geht davon aus, daß der neue Floppy-Befehl im Computer-RAM bei Adresse $9000-$93ff steht und in den gleichen Adreßbereich im Floppy-RAM gehört. Ferner gibt es dem neuen Befehl das Symbol X.

Listing 2 ist der neue Befehl X, der das Floppy-ROM ausliest. Um ihn zu verstehen, sehen wir uns zunächst einmal an, wie der Atari mit seinen Peripheriegeräten kommuniziert.

Sobald der Computer auf der Kommandoleitung einen Low-Impuls gibt (s. Belegung PORTB), wissen die Peripheriegeräte, daß ein sog. Command Frame folgt. Dies sind 5 Byte, die folgende Bedeutung haben:

1. Device I. D.
2. Kommando
3. DAUX
4. DAUX+1
5. Checksumme

Durch Device I. D. erkennt ein Peripheriegerät, ob es gemeint ist (für Diskettenstation 1 ist das z.B. $31). Kommando, DAUX und DAUX+1 sind genau die Bytes, die im Computer bei Adresse $302, $30a und $30b stehen. Sind Device I. D. $31 und die Checksumme in Ordnung, so sendet das Floppy-Betriebssystem eine $41 (Acknowledge) zum Computer, was bedeutet, daß der Befehl verstanden wurde. Nun holt es aus den Tabellen bei $97c0 und $97e0 die Anfangsadresse des entsprechenden Befehls und ruft diesen als Unterprogramm auf.

In DAUX und DAUX+1 ($82, $83) steht somit die Adresse, ab der 256 Byte des ROM ausgelesen werden sollen. Ist DAUX+1 positiv, so wird auf den zweiten 4-KByte-Block umgeschaltet und das oberste Bit von DAUX+1 gesetzt. Jetzt werden 256 Byte aus dem ROM ins RAM kopiert. Danach muß wieder auf den ersten 4-KByte-Block umgeschaltet werden. Nun wird dem Computer signalisiert, daß der Befehl abgeschlossen ist. Danach werden die 256 Byte zum Computer übertragen.

Kommunikation zwischen Computer und Floppy

Das Programm in Listing 3 liest mit Hilfe des neuen X-Befehls das Floppy-ROM aus. Wenn man alle drei Programme als Objekt-File auf Diskette vorliegen hat, so geht man hierzu folgendermaßen vor:

1. Ins DOS
2. Das Programm aus Listing 2 laden (Der neue Befehl)
3. Das Programm aus Listing 1 starten (Floppy wird programmiert)
4. Das Programm aus Listing 3 starten (Floppy-ROM wird in Computer geladen)
5. Den Speicher von $6000-$6fff als DROM1 abspeichern
6. Den Speicher von $7000-$7fff als DROM2 abspeichern

Diese zwei Blöcke müssen anschließend noch disassembliert und bearbeitet werden. Es folgt z.B. auf jeden JSR $9600 und JSR $960a eine 2-Byte-Adresse, die vom Disassembler sicher fälschlicherweise als Befehl interpretiert wird. Auch die Einführung von symbolischen Sprungadressen erleichtert das Verständnis des Floppy-Betriebssystems.

Zum Abschluß dieses Artikels möchte ich noch eine Reihe wichtiger ROM-Routinen und symbolischer Adressen nennen.

In Block 1:

$f000 Sende Puffer zum Computer
$f002 Sende Akku zum Computer
$f040 Neustart
$f1fb Motor aus
$f212 Starte Motor
$f239 Motor an
$f275 Kopf auf Spur 0
$f2ec Kopf auf Spur
$f362 Controler Reset
$f40b Kommando auswerten
$f485 Sende Acknowledge
$f48a Sende Nack
$f48f Sende Complete
$f494 Sende Error
$f499 Empfange 1 Byte vom Computer
$f5bb Lese einen Sektor
$f6a8 Schreibe einen Sektor
$f8d4 Formatiere eine Spur
$fba3 Motorblinken bei Fehler

In Block 2:

$f0f2 Density feststellen
$f37e Lese Spur ein
$f7db Spur und Sektor berechnen

$80 BUSID
$81 DKMD
$82 DAUX
$8d Spur
$99 Pufl
$9a Pufh
$8f Statusr
$B1 Chksum

$9600 Swi jmp(Switch und jmp)
$960a Swi jsr (Switch und jsr)
$96f7 Dsktyp
0: Enhanced Density
1: Double Density
$80: Single Density
$97a0 Ramkmdt
$97c0 Ramkmdl
$97e0 Ramkmdh

Im nächsten Heft soll der Umgang mit den einzelnen FDC-Befehlen beschrieben und nützliche neue Befehle für Ihre Happy vorgestellt werden.

Wer sich nicht die Arbeit machen will, ein eigenes ROM-Listing anzufertigen, kann es auch von mir (gegen Vorauszahlung von 10.– DM) erhalten. Die zwei Blöcke sind in Mac/65-Format abgespeichert.

Stefan Wachter
Haslacher Weg 45
7900 Ulm
Tel. 0731/265303

Listing 1

```
; PROGRAMMIERT DIE FLOPPY MIT
; DEM NEUEN X-BEFEHL, DER IM
; SPEICHER BEI BEFADR STEHEN MUSS
;
DSBI     =   $0300
DDRV     =   $0301
DKMD     =   $0302
DSTA     =   $0303
DPUF     =   $0304
DTIO     =   $0306
DLEN     =   $0308
DAUX     =   $030A
SIO      =   $E459
;
PUF      =   $0600
BEFTAB   =   $9780
BEFADR   =   $9000
;
         *=  $A000
;
; BEFEHLSTABELLE LADEN
;
         LDX # <PUF
         LDY # >PUF
         JSR SETPUF
         LDX # <BEFTAB
         LDY # >BEFTAB
         JSR SETAUX
         LDA #'R
         JSR DOSIO
```

```
;
; LEEREN BEFEHLSPLATZ SUCHEN
;
        LDY #$20
SUCH    LDA PUF,Y
        BEQ LEER
        INY
        JMP SUCH
;
; NEUEN BEFEHL EINFUEGEN
;
LEER    LDA #'X       ; NEUER BEFEHL
        STA PUF,Y     ; X
        LDA # <BEFADR
        STA PUF+$20,Y ; BEFEHLSADR.
        LDA # >BEFADR ; SETZEN
        STA PUF+$40,Y
;
; NEUE TABELLE ZUR FLOPPY
;
        LDA #'P
        JSR DOSIO

;
; NEUEN BEFEHL ZUR FLOPPY
; SCHICKEN
; ( COMPUTER-RAM  $9000-$93FF )
; ( IN FLOPPY-RAM $9000-$93FF )
;
        LDX # <BEFADR
        LDY # >BEFADR
        JSR SETPUF
        JSR SETAUX
PROG    LDA #'P
        JSR DOSIO
        CLC
        LDA DPUF
        ADC #$80
        STA DPUF
        BCC *+5
        INC DPUF+1
        LDA DPUF
        STA DAUX
        LDA DPUF+1
        STA DAUX+1    ; KOMMANDO KANN
        CMP #$94      ; BIS $93FF
        BCC PROG      ; GEHEN
;
; RAMBEREICH IN FLOPPY SCHUETZEN
;
        LDX #$60
        LDY #$60
        JSR SETAUX
        LDA #'H
        JSR DOSIO
;
; FERTIG, FLOPPY PROGRAMMIERT
;
        RTS
```

```
;
SETPUF  STX DPUF
        STY DPUF+1
        RTS
SETAUX  STX DAUX
        STY DAUX+1
        RTS
DOSIO   LDX #0
        CMP #'P
        BNE *+4
        LDX #$80
        CMP #'R
        BNE *+4
        LDX #$40
        STX DSTA
        STA DKMD
        LDA #'1

        STA DSBI
        LDA #1
        STA DDRV
        LDA #2
        STA DTIO
        LDA #128
        STA DLEN
        LDA #0
        STA DTIO+1
        STA DLEN+1
        JSR SIO
        BMI FEHLER
        RTS
;
FEHLER  LDA 710       ;BEI FEHLER
        PHA           ;HINTERGRUND-
        SBC #8        ;FARBE AENDERN
        STA 710
        LDA #0
        STA 20
WARTE   LDA 20        ;WARTEN
        CMP #50
        BCC WARTE
        PLA           ;ALTE HINTER-
        STA 710       ;GRUNDFARBE
        PLA
        PLA
        RTS
;
; STARTADRESSE
;
        *=  $02E0
        .WORD $A000

;
; SENDE 256 DATENBYTES MIT
; CHECKSUMME
;
        LDA # <RAMPUF
        STA PUFL
        LDA # >RAMPUF
        STA PUFH
```

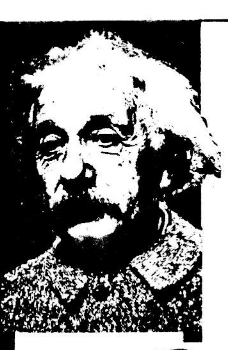
```
          LDY #0
          JSR SENDPUF
;
; ZURUECK ZUM SYSTEM
;
          RTS
```

Listing 2

```
; DIESER BEFEHL ERMOEGLICHT DAS
; DAS AUSLESEN DES FLOPPYBETRIEB-
; SYSTEMS
;
DAUX      =    $82
PUFL      =    $99
PUFH      =    $9A
RAMPUF    =    $8000
;
SENDCPL   =    $F48F
SENDPUF   =    $F503
;
          *=   $9000    ; KOMMANDOADR.
;
          LDA DAUX+1    ; BEI POSITIVEM
          BMI NOSWITCH  ; DAUX+1
          ORA #$80      ; 2. 4K ROM
          STA DAUX+1    ; EINBLENDEN
;
; BLENDE 2. 4K ROM EIN
;
          LDA $FFF9
;
NOSWITCH  LDY #0
LOOP      CPY #$F8      ; AUFPASSEN,
          BCC OK        ; DASS NICHT
          CPY #$FA      ; AUS VERSEHEN
          BCS OK        ; UMGEBLENDET
          LDA DAUX+1    ; WIRD
          CMP #$FF      ; ( LDA $FFF8 )
          BNE OK        ; ( LDA $FFF9 )
          JMP WEITER
;
OK        LDA (DAUX),Y
          STA RAMPUF,Y
WEITER    INY
          BNE LOOP
;
; BLENDE 1. 4K ROM EIN
;
          LDA $FFF8
;
; SIGNALISIERE COMPUTER, DASS
; OPERATION BEENDET IST UND
; DATENBLOCK GLEICH FOLGT
;
          JSR SENDCPL
;
; DAS FLOPPY-ROM BEFINDET SICH
; JETZT IM COMPUTER SPEICHER
; 1. 4K BEI PUF1
; 2. 4K BEI PUF2
;
          RTS
;
DOSIO     LDA #'1
          STA DSBI
          LDA #1
          STA DDRV
          LDA #'X       ; DAS NEUE
          STA DKMD      ; KOMMANDO
          LDA #$40      ; STATUS READ
          STA DSTA
          LDA #2
          STA DTIO
          LDA #0
          STA DTIO+1    ; 256 BYTES
          STA DLEN      ; WERDEN VON
          LDA #1        ; DER FLOPPY
          STA DLEN+1    ; ERWARTET
          JSR SIO
          BMI FEHLER
          RTS
;
FEHLER    LDA 710       ; BEI FEHLER
          PHA           ; HINTERGRUND-
          SBC #8        ; FARBE AENDERN
          STA 710
          LDA #0
          STA 20
WARTE     LDA 20        ; WARTEN
          CMP #50
          BCC WARTE
          PLA           ; ALTE HINTER-
          STA 710       ; GRUNDFARBE
          PLA
          PLA
          RTS
;
; STARTADRESSE
;
          *=  $02E0
          .WORD $5000
```

Listing 3

```
; LAEDT DURCH DEN X-BEFEHL DAS
; FLOPPY BETRIEBSSYSTEM IN DEN
; COMPUTER
; ( 1. 4K NACH PUF1 )
; ( 2. 4K NACH PUF2 )
;
DSBI      =    $0300
DDRV      =    $0301
DKMD      =    $0302
DSTA      =    $0303
DPUF      =    $0304
DTIO      =    $0306
DLEN      =    $0308
DAUX      =    $030A
SIO       =    $E459
;
PUF1      =    $6000
PUF2      =    $7000
;
          *=   $5000
          LDA # <PUF1  ; 1. 4K ROM
          STA DPUF     ; NACH PUF1
          LDA # >PUF1
          STA DPUF+1
          LDA # <$F000 ; NEG. DAUX+1
          STA DAUX     ;  SPRICHT
          LDA # >$F000 ; 1. 4K ROM
          STA DAUX+1   ; AN
;
LOOP1     JSR DOSIO    ; LADE 1. 4K ROM
          INC DPUF+1   ; IN COMPUTER
          INC DAUX+1
          LDA DAUX+1
          BNE LOOP1
;
          LDA # <PUF2  ; 2. 4K ROM
          STA DPUF     ; NACH PUF2
          LDA # >PUF2
          STA DPUF+1
          LDA # <$7000 ; POS. DAUX+1
          STA DAUX     ;  SPRICHT
          LDA # >$7000 ; 2. 4K ROM
          STA DAUX+1   ; AN
;
LOOP2     JSR DOSIO    ; LADE 2. 4K ROM
          INC DAUX+1   ; IN COMPUTER
          INC DPUF+1
          LDA DAUX+1
          CMP #$80
          BNE LOOP2
```



TOPPROGRAMM DES MONATS

1000,-

Zum Topprogramm haben wir in diesem Heft das Programm "XL-TOS" von Marc Ebner aus Gerlingen bei Stuttgart ausgewählt. Es simuliert auf einem Atari 800 XL die GEM-Oberfläche wie beim ST. Damit können jetzt auch die 8-Bit-User so richtig mit GEM werkeln.

Marc Ebner ist 17 Jahre alt und besucht zur Zeit die 11. Klasse am Gerlinger Gymnasium. Seine ersten Erfahrungen sammelte er auf einem 600 XL, den er relativ schnell in Basic programmieren konnte. Danach folgte Assembler und bald darauf C. Im Sommer 1985 ging er dann für ein halbes Jahr auf eine Schule in Amerika und lernte Pascal. Da er sich dort aber mangels nötigem Kleingeld nur einen kleinen Atari leisten konnte, programmierte er sich einfach seine GEM-Oberfläche selbst.

Nach seiner Rückkehr aus den USA begann Marc dann im Februar 1986 auf dem ST zu programmieren. Nach einigen Utility-Programmen folgte die Arbeit an einem Karate-Spiel, das inzwischen fertig geworden ist. Inzwischen programmiert er an einem neuen Spiel, das sich an "The Last Starfighter" anlehnt.

Neben seiner großen Leidenschaft "Computer" geht Marc ganz gern ins Kino. Und was die Zukunft anbelangt, so will er nach dem Abitur Informatik studieren.

